Scaling Up Programming by Demonstration for Intelligent Tutoring Systems Development: An Open-Access Web Site for Middle School Mathematics Learning

Vincent Aleven, Bruce M. McLaren, and Jonathan Sewall

Abstract—Intelligent tutoring systems (ITSs), which provide step-by-step guidance to students in complex problem-solving activities, have been shown to enhance student learning in a range of domains. However, they tend to be difficult to build. Our project investigates whether the process of authoring an ITS can be simplified, while at the same time maintaining the characteristics that make ITS effective, and also maintaining the ability to support large-scale tutor development. Specifically, our project tests whether authoring tools based on programming-by-demonstration techniques (developed in prior research) can support the development of a large-scale, real-world tutor. We are creating an open-access Web site, called Mathtutor (http://webmathtutor.org), where middle school students can solve math problems with step-by-step guidance from ITS. The Mathtutor site fields example-tracing tutors, a novel type of ITS that are built "by demonstration," without programming, using the Cognitive Tutor Authoring Tools (CTATs). The project's main contribution will be that it represents a stringent test of large-scale tutor authoring through programming by demonstration. A secondary contribution will be that it tests whether an open-access site (i.e., a site that is widely and freely available) with software tutors for math learning can attract and sustain user interest and learning on a large scale.

Index Terms—Homework support systems, adaptive educational systems, intelligent tutoring systems, authoring tools.

1 Introduction

Students in middle schools and high schools spend a great deal of time outside of school. If they want to use part of that time solidifying and extending the mathematics they learned in school, however, they will have difficulty finding effective learning activities. After-school tutoring programs are available only on a limited basis and are often not ideal. Professional tutoring services are costly. Of the many Web sites available for math instruction, few are free and offer rich problems with guided learning by doing. Thus, there is a great need for a better math Web site, one where students can "learn by doing," as they engage in rich and challenging math problems. To address this need, we are creating *Mathtutor* (http://webmathtutor.org), an open-access Web site (i.e., a site that is widely and freely available to whoever wishes to use it) for middle school mathematics. The site offers detailed, interactive, step-by-step guidance with problem solving, individualized problem selection, detailed reports of student performance for teachers, parents, etc. On the *Mathtutor* site, students work with intelligent tutoring systems (ITSs), programs that provide step-by-step guidance with problem solving to support students in learning a complex cognitive skill [57], [60]. A first version of the *Mathtutor* site, with limited content, has just been opened to the public. Over the next two years, we will continue to add content to the site and extend its capabilities. Eventually, *Mathtutor* will offer a comprehensive range of middle school math topics (grades 6-8). *Mathtutor* is likely to be useful as supplemental instruction in many contexts, such as after-school tutoring, summer schools, homework, home schooling, classroom instruction, and remediation during the regular school day.

In developing *Mathtutor*, we address an important open issue in research on ITS authoring tools and methodologies. While ITSs have been shown to be very effective in a range of domains [24], [60], they tend to be difficult to build. Many authoring tools have been built to support and simplify ITS development [37], but few have reached a state of maturity, where we can say with confidence that they can support *large-scale* ITS development projects with an extended software life cycle. Tools such as the *Assistments Builder* [21] and *ASPIRE* [34] have been used to build major ITS, but on the whole, large-scale projects using dedicated ITS authoring environments are still quite rare. Such projects may bring up maintainability and scalability issues that do not necessarily come up in smaller scale projects, so reports

- *V. Aleven is with the Human-Computer Interaction Institute, Carnegie Mellon University, 3613 Newell Simon Hall, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891. E-mail: aleven@cs.cmu.edu.*
- *B.M. McLaren is with the Human-Computer Interaction Institute, Carnegie Mellon University, 2617 Newell-Simon Hall, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891, and the Competence Center for e-Learning, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany. E-mail: bmclaren@cs.cmu.edu.*
- *J. Sewall is with the Human-Computer Interaction Institute, Carnegie Mellon University, 2617 Newell-Simon Hall, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891. E-mail: sewall@cs.cmu.edu.*

Manuscript received 8 Jan. 2009; revised 23 Mar. 2009; accepted 4 Apr. 2009; published online 5 May 2009.
For information on obtaining reprints of this article, please send e-mail to: lt@computer.org, and reference IEEECS Log Number TLTSI-2009-01-0002.
Digital Object Identifier no. 10.1109/TLT.2009.22.

on the experience of using ITS authoring tools in large-scale projects are highly desirable. The current paper is such a report; it describes an early stage of a large-scale ITS project.

In creating *Mathtutor,* we use a novel technology for ITS called *example-tracing tutors,* developed in our lab. These tutors support key behaviors that make ITS effective, such as step-by-step hints and feedback during problem-solving activities [57], but they are much easier to build than typical ITS. Example-tracing tutors derive, from the Cognitive Tutor technology [24], [25], an approach to ITS that has been shown to be highly effective in improving students' math achievement levels [24], [36], [45], [54]. Cognitive Tutors are rooted in cognitive theory [3] and cognitive modeling. Like most other ITS, they support learning by doing—students learn by tackling complex problems for which the tutor provides step-by-step guidance [57], typically, problems of a recurrent type (e.g., solving algebraic equations) rather than "one-of-a-kind" problems, and often problems that are amenable to multiple solution strategies. Cognitive Tutors operate using a rule-based cognitive model of student problem solving, a computer simulation of student thinking. The tutor uses the model to evaluate student problem-solving behavior, track individual students' skill development over time, and select problems on an individual basis [4], [22].

Example-tracing tutors maintain both the grounding of Cognitive Tutors in cognitive theory as well as their essential tutoring behaviors. The underlying technology, and the authoring process used to construct the tutors, however, are very different. Example-tracing tutors have two main advantages over Cognitive Tutors. First, the Cognitive Tutor Authoring Tools (CTATs) [1], [23] make it possible to build example-tracing tutors faster and more economically than traditional development time estimates for ITS [37], without requiring advanced computer programming skills. Whereas typical ITS development tends to require advanced AI programming skill, example-tracing tutors can be built relatively easily by nonprogrammers, through "programming by demonstration" [29], [38]. Second, CTAT makes it possible to deliver intelligent tutors on the Web, which is now the preferred mode of delivery for ITS [2], [13], [16], [18], [32], [33], [35], [46], [53], [58], [61], although there are still architectural issues to resolve in order to robustly support intelligent tutors on the Web [43]. Our claim is not that example-tracing tutors are always a preferred option over Cognitive Tutors. They may not be, for example, for tasks that have many operators whose results differ depending on the state in which they are applied. However, there is a large class of task domains for which example-tracing tutors are well suited.

Whereas Cognitive Tutors use a rule-based cognitive model of the targeted problem-solving skills to monitor student problem-solving behavior, example-tracing tutors evaluate student problem-solving behavior with reference to specific examples of problem solutions. An essential step in the process of authoring an example-tracing tutor is for an author to *demonstrate* the problem-solving behaviors for which the tutor is to provide guidance, as well as to demonstrate common student errors. CTAT records the demonstrated problem-solving steps in a *behavior graph.* The author *generalizes* the graph among other ways by attaching *ordering constraints* and *formulas*, described further below. At tutoring time, an example-tracing tutor evaluates student problem-solving behavior by flexibly comparing it against the generalized behavior graph. Thus, the tutor is capable of recognizing as correct student behavior not just the fixed sequences of problem-solving steps recorded in the behavior graph, but many variants as well. The resulting tutoring behaviors include many features characteristic of ITS, as catalogued by VanLehn [57]. CTAT is not unique among ITS authoring tools in supporting authoring without programming. The *Assistments Builder* [21], *ASPIRE* [34], and the *Task Tutor Toolkit* [41] also support authoring of tutors of varying levels of sophistication without requiring programming. The *Assistments Builder* supports simpler tutors than CTAT through Web-based authoring. *ASPIRE* supports a different type of tutors, constraint-based tutors, but its GUI builder is much simpler than CTAT's. The *Task Tutor Toolkit* offers a programming-by-demonstration approach not unlike that of CTAT, but CTAT has a more powerful tutor engine and places more emphasis than the *Task Tutor Toolkit* on efficient authoring and maintenance of large sets of isomorphic or near-isomorphic tutor problems.

CTAT-built tutors have been demonstrated to be fully robust for use in real educational settings over a wide range of projects. So far, at least 26 research studies have used CTAT-built tutors, making CTAT the most widely used ITS authoring tool that we know of. These projects provide evidence that programming-by-demonstration techniques can support highly cost-effective authoring of real-world ITS, compared to historic estimates of authoring efficiency [1]. However, these projects, while being "real-world," tended to have somewhat limited scope in terms of curricular breadth (e.g., 1 or 2 hours of instruction), and the number of students that used the CTAT-developed tutors (e.g., 30-100). The current project, by contrast, involves much greater curricular breadth than prior projects, as well as longer and more independent real-life use, iterative refinement of tutors, and—we anticipate—at least an order of magnitude more users.

While a programming-by-demonstration approach unquestionably simplifies the ITS authoring process, it has not been fully proven that this type of process can support the development, iterative refinement, and maintenance of a *large* set of tutors. When not applied carefully, CTAT's programming-by-demonstration process can lead to extensive problem-specific authoring, with repeated structure created across behavior graphs for different problems. This duplication of structure would be detrimental to maintainability because any needed changes must then be propagated across many similar or near-isomorphic problems. This risk must be offset by tools or tool features that make it easier to establish and maintain consistency across problems and problem types. CTAT has a number of such features. A key question is whether we have struck a good balance between ease of authoring and maintainability. As discussed below, we believe that we are on the right track, but the proof will be in whether we can grow *Mathtutor* to large scale.

A second aspect of scale concerns the requirement that a Web-based ITS architecture (such as *Mathtutor*'s) must support sophisticated tutoring behaviors even when hundreds, or even thousands, of users work on the system simultaneously. We investigate whether a "thin client" solution can handle such a high load effectively. In this approach, the tutor interface runs on the client but the tutor engine runs on the server, with frequent communication between them. Some Web-based ITS architectures have opted for the same basic division of functionality (e.g., [2], [32], [34]) while others have placed the tutor engine on the client [46]. It is fair to say that this issue is still open [43], particularly as ITS use scales up, and there may not be a single best solution. The thin client solution may be especially appropriate in an ITS *authoring* architecture (such as CTAT's on which *Mathtutor* is built), which aims to support many different ITS configurations in a variety of settings, specifically when the same tutor engine is combined with many different tutor interface options.

Key anticipated contributions of the *Mathtutor* project are, first, a demonstration that programming by demonstration can scale up and support a large tutor development project. Also, we will learn whether a thin client approach is feasible to support a large-scale ITS Web site. Beyond these technical contributions, the *Mathtutor* project will provide insight into whether an open-access site, for mathematics learning with ITS, appeals to the general public and will be an effective way of improving students' mathematics achievement in a variety of contexts. As mentioned, we do not have full answers to these questions yet. The experience in the coming two years will be interesting and informative regarding these questions.

In this paper, we first describe the main goals and foundation of the *Mathtutor* project and present examples of tutors fielded on the site. We then describe the process of programming by demonstration by which these tutors are built and sketch out the main algorithm in the example-tracing tutor engine. We also motivate the key architectural choices in the *Mathtutor* site, including the decision to have a thin client. We then compare and contrast *Mathtutor* with other Web sites for math learning. Finally, we discuss our initial experience with regard to the key scalability and maintainability issues raised above.

2 The Foundation for *Mathtutor:* Cognitive Tutor Math Curricula

Cognitive Tutor curricula for high school mathematics have been very successful in real educational settings. These courses are aligned with the (nonbinding but influential) standards defined by the National Council of Teachers of Mathematics (NCTM) [40], a large US organization concerned with mathematics education. Students in these courses use Cognitive Tutor about 40 percent of their classroom time. In 1999, the Cognitive Tutor Algebra I curriculum was designated by the US Department of Education as one of five "exemplary curricula." The Cognitive Tutor curricula have been shown to improve student learning significantly, compared to other math curricula (e.g., [24], [36], [45], [54]). The high school curricula are in use in over 2,600 schools across the USA and are being disseminated by Carnegie Learning, Inc., a Pittsburgh-based company founded by Carnegie Mellon University.

As we develop the *Mathtutor* site, we draw heavily from an existing sequence of 6th, 7th, and 8th grade Cognitive Tutor curricula for middle school math. These curricula were developed in our research group and are based on the same principles as the high school courses. They have been piloted and incrementally improved in a number of schools over several years. The middle school curricula also reflect NCTM [40] curriculum, teaching, and assessment standards. They target and develop five content strands across the three-year sequence as follows:

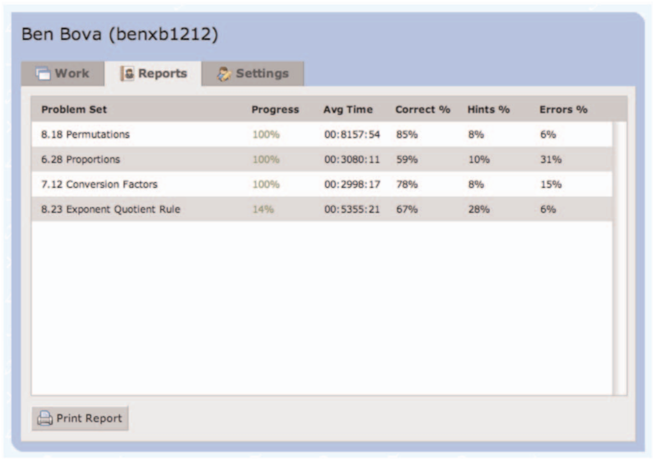
1. numbers and operations,
2. algebra,
3. data analysis,
4. geometry, and
5. measurement.

Across the three middle school courses, 66 Cognitive Tutor lessons comprising over 1,000 problems have been developed. Text and material development have been guided by research on students' mathematical thinking [9], [11], [27], [49], [46] and formative evaluations of text and tutor activities [10], [12], [20]. Nine end-of-course summative evaluations of the Cognitive Tutor middle school mathematics courses have been completed in two partner suburban school districts (Corbett, personal communication). Each of these evaluations compared learning outcomes for students in the Cognitive Tutor courses with students in corresponding traditional courses, with respect to two types of tests: a test consisting of standardized test questions drawn from various sources and one consisting of open-ended problem-solving questions. The Cognitive Tutor students scored higher on all nine of the standardized tests; seven of the individual comparisons were statistically significant or marginally significant. The same result was obtained for the problem-solving tests, providing strong preliminary evidence of the efficacy of the curricula and the tutors within them.

The *Mathtutor* site will eventually cover the same five content strands across a three-year sequence, in the same breadth and depth. The scope and sequence of the material remains largely the same. Further, in creating tutors for *Mathtutor* we reuse existing problem types and specific problems, as much as possible, reimplementing them as example-tracing tutors using CTAT to permit Web access. This reimplementation effort is necessary because the original Cognitive Tutors do not run on the Web.

3 *Mathtutor* Overview and Context of Use

As mentioned, the *Mathtutor* site is open to anyone: any teacher or student can sign up for the site. Students using the site learn math by doing problems with guidance from the ITS. Teachers who sign up for the site can create class lists, assign work to an entire class or an individual student, and view reports of their students' progress (Fig. 1). The content on the site is divided into *problem sets,* usually consisting of about a dozen multistep problems. Teachers can select from the existing problem sets to make assignments to their students. A future version of *Mathtutor* will provide facilities


Ben Bova (benxb1212)

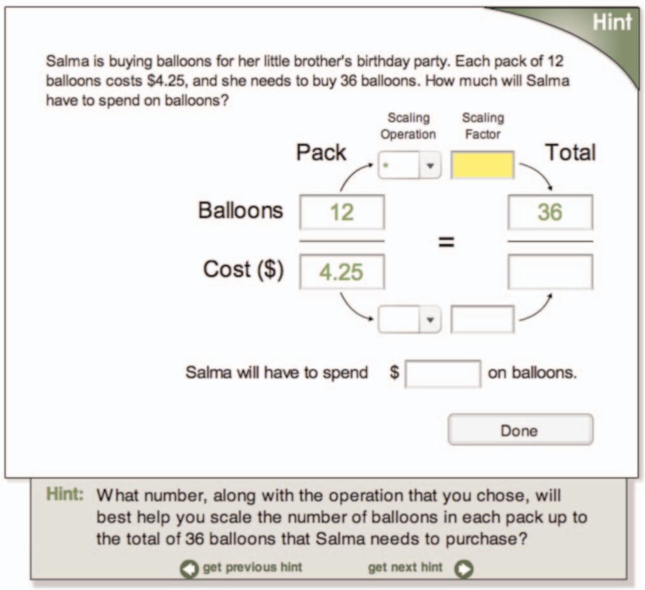
Problem Set	Progress	Avg Time	Correct %	Hints %	Errors %
8.18 Permutations	100%	00:8157:54	85%	8%	6%
6.28 Proportions	100%	00:3080:11	59%	10%	31%
7.12 Conversion Factors	100%	00:2998:17	78%	8%	15%
8.23 Exponent Quotient Rule	14%	00:5355:21	67%	28%	6%

Fig. 1. *Mathtutor* report showing detailed performance data for one student; this type of report is available to teachers and students.

for teachers to easily *assemble* problem sets for their students, by selecting problems from other sets available on the site. The problem sets are indexed by state standards. Currently, only the Pennsylvania standards are supported, but we plan to index using standards from other states, and national standards such as the NCTM Focal Points [39].

Mathtutor generates reports of student performance, capitalizing on the fact that the tutors, as part of their regular operation, evaluate student answers. Currently, it supports three types of reports: 1) a report detailing the progress of a class of students through the assigned sequence of problem sets, 2) a report detailing the performance of a class of students on a given problem set, listing the time spent on the problem set and the frequency of correct steps, errors, and hints, and 3) a report detailing a single student's performance on the assigned problem sets, listing the same statistics for each problem set as those in the second report (Fig. 1). Future reports will provide more detail with respect to students' performance on individual problems and the individual skills targeted in the instruction, and will also break down each student's performance relative to specific standards. These reports have the potential to help teachers focus their instructional efforts on areas in which students could benefit the most (see also [14], [15]), an advantage that has been claimed for the *Assistments* Web site [46] described below, which also fields tutors that enable detailed assessment of students.

A first version of *Mathtutor* is being pilot-tested in the context of after-school tutoring programs, which are required in the US under the "No Child Left Behind" act, federal legislation aimed at improving the performance of US primary and secondary schools (http://www.ed.gov/nclb/overview/intro/factsheet.html). The Web site is expected to be beneficial in this context, as supplemental instruction, for a number of reasons. First, human tutors or teachers working with students after school could use the Web site to supplement and extend their own efforts with the students. In the common case, where there are many more students than human tutors, the tutors could set their several students at work on problems on the site, possibly giving each student his or her own problem sets to work on. The *Mathtutor* site may be especially helpful when


Salma is buying balloons for her little brother's birthday party. Each pack of 12 balloons costs $4.25, and she needs to buy 36 balloons. How much will Salma have to spend on balloons?

Salma will have to spend $ on balloons.

Hint: What number, along with the operation that you chose, will best help you scale the number of balloons in each pack up to the total of 36 balloons that Salma needs to purchase?

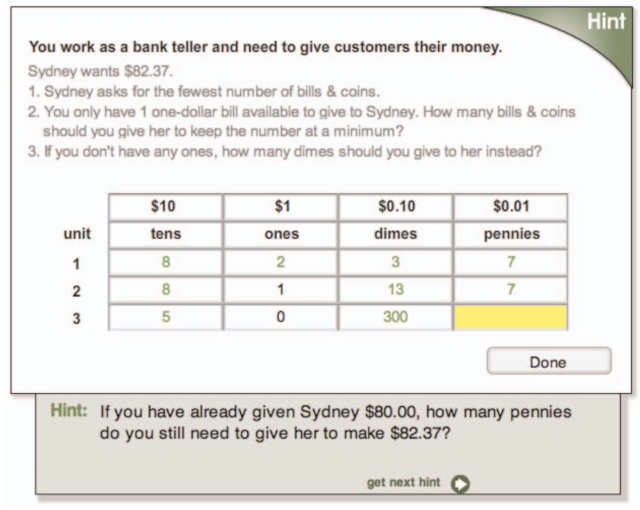
Fig. 2. Mathtutor problem dealing with proportional reasoning.

after-school programs are led by teachers who are not math teachers, as happens far more frequently than would be ideal. Finally, the *Mathtutor* site might facilitate communication between a student's regular math teacher and the after-school teacher/tutor. In particular, tutor and teacher could periodically get together to view *Mathtutor* reports of the student's performance and plan further activities.

Although we are piloting *Mathtutor* primarily in the context of after-school and math remediation activities, we envision that the site will be useful in many other contexts as well. For example, teachers could incorporate the use of the site into their regular classes. They could assign homework on the site or use *Mathtutor* materials for in-class exercises or quizzes. Further, the site will offer help to students doing math outside of regular school hours or after-school programs, whether for remedial purposes or as part of their own efforts to move ahead of the regular math instruction in school. The site could also aid parents trying to help their children with math. In a future version, parents will be able to view reports about their children's performance. Finally, teachers and parents could use the site to brush up on their own math, so they are better able to help their students and children.

4 Examples of Tutors

In this section, we illustrate some of the *Mathtutor* problem-solving exercises and highlight some features of the tutors. Fig. 2 shows a *Mathtutor* problem for 7th-graders dealing with early proportional reasoning, designed to help students acquire a conceptual understanding of proportionality. The reader can try out this particular tutor by going to http://webmathtutor.org. The interface of the tutor has been carefully designed to make the thinking steps visible [4]. Here, the student must identify the operation that leads from the number of balloons in one pack (12) to the desired number of balloons (36) so that he or she can then apply that same operation to the cost of one pack of balloons ($4.25). The



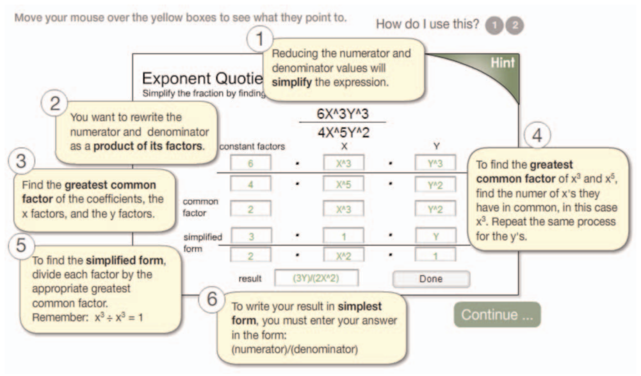
You work as a bank teller and need to give customers their money.

Sydney wants $82.37.
1. Sydney asks for the fewest number of bills & coins.
2. You only have 1 one-dollar bill available to give to Sydney. How many bills & coins should you give her to keep the number at a minimum?
3. If you don't have any ones, how many dimes should you give to her instead?

	$10	$1	$0.10	$0.01
unit	tens	ones	dimes	pennies
1	8	2	3	7
2	8	1	13	7
3	5	0	300	

Hint: If you have already given Sydney $80.00, how many pennies do you still need to give her to make $82.37?

Fig. 3. *Mathtutor* problem for decimal place value.



Move your mouse over the yellow boxes to see what they point to.

1 Reducing the numerator and denominator values will **simplify** the expression.

2 You want to rewrite the numerator and denominator as a **product of its factors**.

3 Find the **greatest common factor** of the coefficients, the x factors, and the y factors.

4 To find the **greatest common factor** of x^3 and x^5, find the numer of x's they have in common, in this case x^3. Repeat the same process for the y's.

5 To find the **simplified form**, divide each factor by the appropriate greatest common factor. Remember: $x^3 \div x^3 = 1$

6 To write your result in **simplest form**, you must enter your answer in the form: (numerator)/(denominator)

Fig. 4. "Intro screen" for one of the problem sets on *Mathtutor;* the annotations actually appear on the site as students' view this problem set (i.e., they were not added for this paper).

tutor provides feedback: correct problem steps are highlighted in green. The tutor offers hints upon the student's request at every step along the way. The panel at the bottom of the tutor interface provides help with the current step.

As discussed further below, example-tracing tutors are capable of following along with the student in problems that allow for multiple solution paths, which simpler tutors cannot do. The decimal place value tutor shown in Fig. 3 illustrates this capability. It has a large space of possible solutions, since there are many ways of breaking down $82.37 into tens, ones, dimes, and pennies (even when, as in this problem, the number of ones is given). The solution shown in line 3 is perhaps not one that many students would produce, but the tutor must be able to recognize it anyway, and be able to provide meaningful hints when a student decides to follow this strategy.

The problem sets and tutors we are creating have two novel features that are desirable especially when the users are not in a classroom context, but work by themselves, and therefore, need to be able to come up to speed with the *Mathtutor* problem sets by themselves without the help of peers or teachers. First, we have added an introductory screen to each problem set, with brief explanations of how to approach the type of problems in the set (Fig. 4). The introductory screen shows the tutor interface annotated with text balloons. When the student mouses over a balloon, the relevant part of the tutor screen is highlighted.

Second, we are developing worked examples: at the start of each problem set, the first few problems have worked-out steps. Educational psychology and the learning sciences literature indicate that students learn more robustly when worked examples supplement problem-solving practice [6], [17], [42], [47], [56]. Various studies involving Cognitive Tutors and CTAT-built tutors have also shown that worked examples can be an effective supplement to problem-solving practice [31], [52], [55]. The steps in our examples are faded in a backward manner, meaning that initially the later steps in the given problems are left open, and the earlier steps are worked out, and then gradually, in later problems, more steps are left open, until entire problems are open for the student to solve. This method has been found to be a very effective way of transitioning from worked examples to problem-solving practice [7], [48].

5 CTAT: Authoring Example-Tracing Tutors "By Demonstration"

The tutors on the *Mathtutor* site are example-tracing tutors implemented with the Cognitive Tutor Authoring Tools (CTATs) [1], [23]. As previously discussed, these tutors share many of the essential behaviors of Cognitive Tutors, but are easier to build. They utilize specific examples of problem-solving behaviors to provide tutoring, rather than a rule-based cognitive model of student problem solving. (The fact that the tool suite is named *Cognitive Tutor* Authoring Tools reflects the fact that it was originally created to support the development of regular Cognitive Tutors. In addition to tools for developing Cognitive Tutors, however, the suite includes tools for creating example-tracing tutors. These tools were added after the tool suite had been given its name.) In this section, we describe the process of authoring an example-tracing tutor with CTAT. We highlight the features within CTAT that enhance the maintainability of example-tracing tutors, which will be key to the large-scale development of such tutors for the *Mathtutor* project. We also sketch the algorithm used by example-tracing tutors to evaluate student solution steps. For an in-depth treatment, the reader is referred to [1].

CTAT has been used extensively and the accumulated experience indicates that building example-tracing tutors with CTAT is significantly less costly and time-consuming than building regular Cognitive Tutors. Close to 400 users have built tutors with the CTAT tools, mostly in workshops, tutorials, and courses, but also for research and development projects. Tutors built with CTAT have been used in university, high school, and middle school classrooms as part of regular course assignments and experimental interventions. At the time of this writing, 26 research studies have used CTAT tutors in real educational settings. In these projects, the development of tutors with CTAT is three to four times more efficient than historical estimates in the literature [1], which indicate that creating one hour of instruction with an ITS takes 200-300 hours [37]. Factoring in the lower cost of authoring, since advanced programming skill is not required, ITS development with CTAT

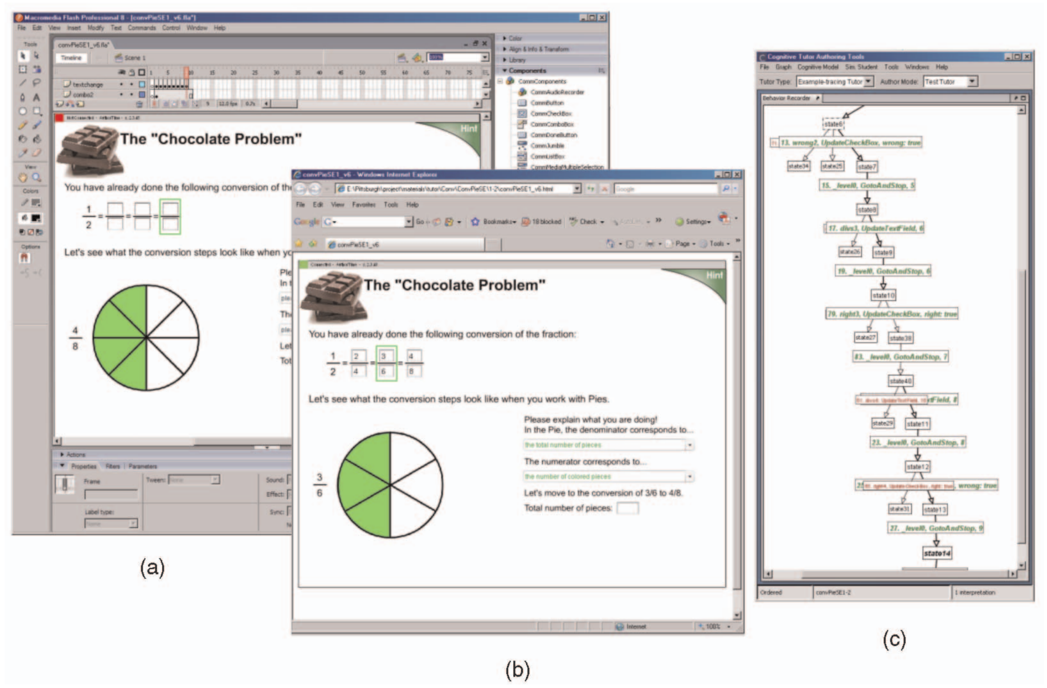

The "Chocolate Problem"
You have already done the following conversion of the fraction:
Let's see what the conversion steps look like when you work with Pies.
Please explain what you are doing!
In the Pie, the denominator corresponds to...
The numerator corresponds to...
Let's move to the conversion of 3/6 to 4/8.
Total number of pieces:
Hint
(a)
(b)
(c)

Fig. 5. Using CTAT to create an example-tracing tutor through programming by demonstration. (a) After creating an interface through drag-and-drop techniques, without programming, (b) an author runs the interface and demonstrates correct and incorrect behavior. (c) CTAT records the steps in a behavior graph and later uses this graph, duly generalized by the author, as a model to evaluate student behavior.

appears to be four to eight times more cost effective than the development of Cognitive Tutors and other types of non-CTAT ITS [1].

5.1 Developing an Example-Tracing Tutor

Creating an example-tracing tutor with CTAT involves five main phases. First, the author builds a user interface for a particular problem type for which tutoring is to be provided, carefully designed to make thinking visible [4]. A typical interface lays out the problem-solving steps for the given problem type. The interface is built through drag-and-drop techniques using an off-the-shelf tool such as Netbeans or Flash (Fig. 5a). The author selects "tutor-enabled interface components" from a palette (the top-right panel in the Flash IDE) and drags them into the desired positions on a central canvas. These components, which were written (by us) in ActionScript and imported into the IDE, communicate with the rest of CTAT. Where tutors require behaviors that cannot be realized with the existing set of tutor-enabled components, ActionScript programming is needed to create new ones. In addition to static "form filling" interfaces, CTAT supports "dynamic interfaces" that can change during the interaction with the student. The tutor controls the interface changes, which can, therefore, be contingent on the state of problem solving by the student. As described below, dynamic interface changes can be authored without programming. For example, at specific points in the problem-solving process, a tutor may hide or show particular interface components, or display new values within given components. For more extensive layout changes, an author can create a Flash interface with multiple frames (essentially, separate layouts containing different interface components) and make the tutor switch from one to the other at specific points in the interaction. Dynamic interfaces are useful in ITS for many purposes, including dynamic scaffolding, dynamically linking of problem-solving representations, and careful management of screen real estate [1]. For example, in the fractions conversion problem in Fig. 5, the tutor (at tutoring time) initially displays a pie divided into six slices, but at an appropriate point in the interaction sequence, the tutor replaces the image by one in which the pie is divided into eight slices, to illustrate the equivalence of 3/6 and 4/8.

In the second phase of authoring, the author runs the interface (Fig. 5b) and demonstrates correct and incorrect problem-solving behavior. CTAT's Behavior Recorder tool records each step in a behavior graph (Fig. 5c). Each link in the graph represents a correct or incorrect step. If there are multiple correct solution strategies, they can be recorded as separate paths in the graph. To record a new path, the author can "back up" to an earlier point in the graph by clicking on one of the recorded states and resume recording under that state. The author can also demonstrate common student errors and mark them as such in the behavior graph. The Behavior Recorder displays them with a label containing red font. Eventually, after the author has completed all five authoring phases and the graph is ready to be used for tutoring, the graph serves as a model for student behavior: the runtime tutor flexibly compares student actions to the graph and provides feedback on the correctness of the student actions.

As a key step in authoring dynamic interfaces without programming, an author can mark certain links in the graph as representing "tutor-performed actions," which (at tutoring time) the tutor executes when the student's solution reaches the state from which they emanate. Typical tutor-performed actions are to show or hide widgets, display a particular value in a widget, or to move to a different frame in the Flash-based tutor interface.

As the third part of tutor construction, the author associates skills, hints, and feedback messages with the steps represented in the behavior graph. To do this, the author clicks on the link labels and fills in pop-up forms soliciting the information. Typically, an author provides multiple levels of hints for each step. Earlier hint levels may identify a problem-solving principle, later hint levels may discuss how to apply it in the given problem. The last hint level may give the specific step to be performed. At tutoring time, hints are presented upon the student's request. The skill labels that are provided as annotations on links typically reflect a fine-grained decomposition of the skills involved in solving a problem (e.g., "find the least common denominator"). The tutor uses the skills to identify and report student progress in mastering particular skills, a capability briefly discussed earlier.

Fourth, the author can generalize the graph to indicate that a wider range of student behavior is to be recognized as correct beyond just the fixed value(s) and sequence(s) of steps recorded in the graph. An author can generalize a behavior graph by specifying ordering constraints on steps or by specifying formulas that describe how problem steps are related. These generalization features enhance the maintainability of example-tracing tutors by keeping down the number of paths in behavior graphs.

In those problems in which (as is typical) there are only fairly loose constraints on the order in which solution steps are to be performed, authors either can specify that any step order within the graph is fine, or they can group steps, nesting groups inside other groups where appropriate, and then specify on a per-group basis whether the student must strictly follow the demonstrated order or can perform the steps in any order. Furthermore, an author can set lower and upper bounds on the number of times a particular step needs to be executed in order for a problem to be considered fully solved. By setting these limits to 0 and 1, for example, an author essentially marks steps as optional.

Authors can also generalize their behavior graphs by attaching "formulas" to links in the graph, similar to Excel formulas, which specify how steps depend on each other. The formulas involve mathematical and string functions. For example, in the tutor for decimal place value shown in Fig. 3, formulas are used to specify that the number of tens, ones, dimes, and pennies entered add up to the given dollar amount. Formulas can dramatically reduce the number of paths in a graph, compared to having to enumerate all paths corresponding to the input described by the formula, and thereby, significantly enhance the maintainability of example-tracing tutors. Formulas represent a level of end-user programming that, as the success of Excel proves, is feasible for people not trained in software engineering. When further functions are needed, besides those currently supported, they can be added with limited Java programming.

As an optional fifth step in the development of example-tracing tutors, in cases in which authors want a number of similar exercises (as is often the case when building tutors), they can avoid the work of demonstrating solutions to each individual problem by employing a "Mass Production" feature. In this approach, an author first creates a behavior graph in the usual manner, tests it, and once it is fully debugged, turns it into a template by replacing problem-specific values and hints with variables. Next, CTAT generates an Excel table with a row for each variable; authors then fill in the table's columns with problem-specific values for each variable, one column per problem. In a final merge step, a behavior graph is created for each column by substituting the variable values into the template. In our experience, the Mass Production process significantly reduces authoring time and enhances the consistency and maintainability of the resulting tutors, as discussed further below. Mass Production is useful even with very small numbers of isomorphic or near-isomorphic tutor problems created from the same template (e.g., 2 or 3). Editing a spreadsheet is faster and decidedly less error-prone than demonstrating solutions over and over. Also, most authors know how to use common spreadsheet tools, such as Excel. Finally, keeping tutors consistent is easier this way.

5.2 The Example-Tracing Algorithm

CTAT's example tracer functions as "step analyzer" in VanLehn's [57] terminology, the module that evaluates whether a given problem-solving step by the student is appropriate and correct, given the current state of problem solving. Although it is beyond the scope of this paper to provide the full details of the example-tracing algorithm, a brief sketch follows. Two broad requirements are that, first, in order to complete a problem, a student must complete a single start-to-finish path through the behavior graph for that problem (i.e., must "traverse" all nonoptional links on such a path, in an order that observes any ordering constraints specified in the graph; as mentioned, these constraints are defined in the form of nested groups of links that are either ordered or unordered). Second, the example tracer does not let the student change his or her mind about the path that he or she takes through the problem. That is, once a student is one more step onto a certain path, all paths that do not match the same steps are ruled out. Any paths that *do* match the same steps remain viable, however, as alternative interpretations of student behavior. As a consequence, steps that have been accepted as correct by the tutor remain correct for the duration of the problem.

More specifically, in order to perform its function as a step analyzer, the example tracer maintains one or more *interpretations* of the students' problem-solving behavior. Each interpretation (at least at a conceptual level) consists of 1) a start-to-finish path through the behavior graph for the given problem and 2) a record of the links on that path that have been matched by student input, together with the student actions that matched the links. At any point in time, all interpretations *must* involve the same set of matched student actions; this set comprises *all* student actions within the current problem that the tutor has so far accepted as correct. When there is new student input, the example tracer evaluates that input by considering whether it can *extend* any of the existing interpretations. To do so, it considers for each interpretation whether the input matches any nonmatched link within that interpretation, subject to the ordering constraints specified in the given behavior graph. As part of this step, the example tracer computes values specified by formulas on links. If no such matching link can be found, the input is rejected and the tutor gives

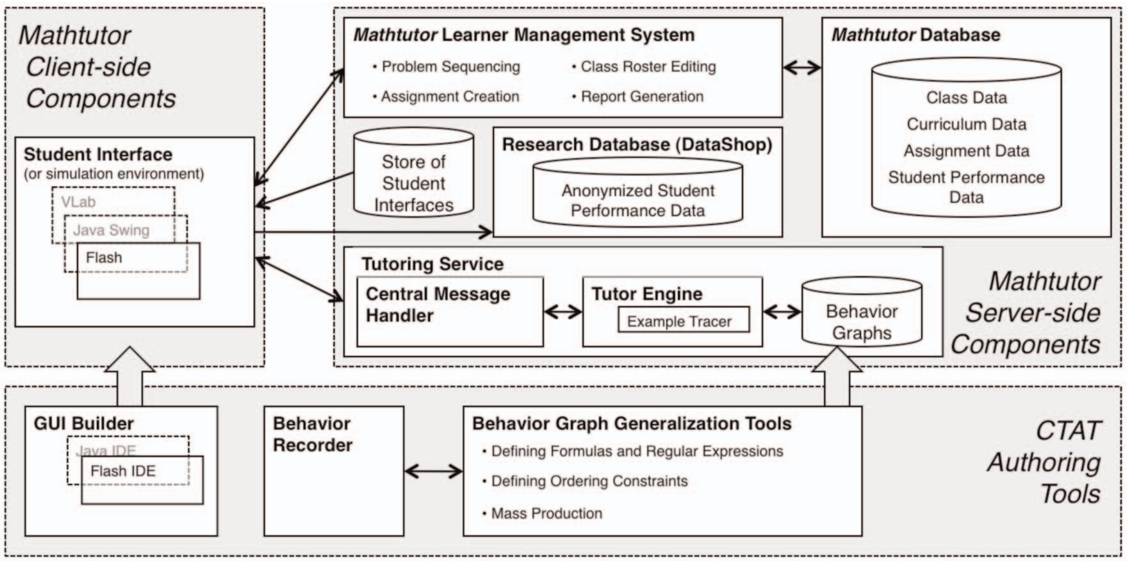


Fig. 6. Mathtutor architecture.

negative feedback to the student. The set of interpretations remains the same, reflecting the fact that the state of problem solving has not been advanced. If, on the other hand, student input *does* match a previously nonmatched link in at least one of the existing interpretations, while also observing the ordering constraints specified in the behavior graph, then the input is accepted, and the set of interpretations is updated as follows. First, all interpretations in which a previously nonmatched link is matched are extended, by recording the newly matched link in their set of matched links. Second, any interpretations that do not have a matching link are dropped. These interpretations are no longer consistent with student behavior, and therefore, must be ruled out. Finally, if a student action matches multiple links within any given interpretation, then that interpretation is split into multiple interpretations, one for each match. The student is done with a problem when all nonoptional links within any single interpretation have been matched in the manner described above.

Example-tracing tutors satisfy, but also go beyond, VanLehn's [57] minimum criterion for being an ITS: they have an "inner loop," meaning that they provide step-by-step guidance with problem-solving activities through feedback messages and next-step hints. Example-tracing tutors go beyond this minimum definition in a number of important ways [1]. First, they are capable of following students along multiple solution paths. Further, CTAT's formula mechanism enables example-tracing tutors to recognize problem solutions, where later steps depend on earlier steps in intricate ways, even when it would be prohibitively difficult to enumerate all possible ways in advance. (The tutor in Fig. 3 illustrates this capability.) Finally, example-tracing tutors are capable of maintaining multiple interpretations of student behavior in situations, where a problem-solving step cannot uniquely be interpreted until later steps have been seen. As a result, example-tracing tutors fluidly follow student strategies even in the face of temporary ambiguity as to what strategy they are following, without the need to ask the student disambiguation questions, and without the risk of prematurely ruling out an interpretation of student behavior that later turns out to have been correct. Such a response would have the undesirable consequence that the tutor would force the student off of a valid solution path they were following. Thus, while example-tracing tutors are built using dramatically simplified authoring technology, compared to their direct ancestors, Cognitive Tutors [5], and to many other ITS, they are by no means simple tutors.

6 *MATHTUTOR* ARCHITECTURE

The *Mathtutor* architecture instantiates CTAT's Web-based architecture, described in [1]. Use of this architecture makes it possible to run example-tracing tutors on the Web, a key motivation for reimplementing the existing Cognitive Tutors for middle school mathematics as example-tracing tutors. As mentioned, a central characteristic of this architecture is that it supports a thin client, meaning that a minimum amount of functionality runs on the student's machine. In this section, we describe the main components of the *Mathtutor* architecture and illustrate the data flow among these components. We also motivate several important architectural choices we have made, in particular, the decision to have a thin client.

6.1 *Mathtutor's* Main Components

The main components of *Mathtutor* (Fig. 6) are the following:

- *The CTAT authoring tools* (Fig. 6, bottom), including the Behavior Recorder and tools for generalizing behavior graphs.
- *GUI builder tools*, such as the Flash IDE is used to create interfaces; we imported ActionScript-based components into the Flash IDE.
- *The Mathtutor Server-Side Components* (top-right of Fig. 6). These server-side components include:
    - *Tutoring Service,* the module that runs the example-tracing tutor engine;

  - *Mathtutor Learner Management System,* used by teachers to select, sequence, and assign problems, edit class rosters, and generate reports; used by students to get problems;
  - Several data stores, including the tutor interfaces; the *DataShop* research database (see below); and the *Mathtutor* Database, which contains class, curriculum, assignment, and student performance data.
- *The Mathtutor Client-Side Components* (top-left of Fig. 6). These are the *Mathtutor* user interfaces that run on the student machines, after being downloaded from the server. Each is designed to support a specific problem type.

The *DataShop,* drawn in Fig. 6 amid the other server-side components, is not part of *Mathtutor* or CTAT proper; rather, it is a separate and publicly available facility for storing and analyzing anonymous log data of student-tutor interactions for research and development purposes. See http://pslcdatashop.org.

6.2 Data Flow among Components at Runtime

In the following paragraphs, we trace the flow of data among the components in Fig. 6 during the workflow of a typical teacher and student using the site. Assume that a teacher, call her Jane, brings up the *Mathtutor* Web site on her browser, creates a class by providing a list of students (possibly copy-and-pasted from a spreadsheet), and assigns a sequence of problem sets to the class. *Mathtutor*'s Learner Management System (LMS) stores the students, class rosters and assignments in the *Mathtutor* Database, which also contains information about the problem sets available on the site.

Next, Fred, a student in Jane's class, brings up the *Mathtutor* Web site in his browser and logs in. The LMS retrieves Fred's class(es) and assignment(s) from the Database and lists them on the screen. After Fred selects one of the open assignments, the LMS presents the first open problem within that assignment. To do so, the LMS retrieves from the Database the name of the Behavior Graph for this first open problem and the name of the Flash program (typically called a "movie") that implements the tutor interface for this problem. It then generates the HTML needed for the browser on Fred's machine to invoke the Flash Player in a special-purpose frame. The Flash Player retrieves the Flash movie from the Store of Student Interfaces on the server and launches it. (Within the thin client approach, this is the only program download required.) The Flash movie first sends the behavior graph name and other parameters to the Tutoring Service, the server-based module that runs the tutor engine for example-tracing tutors. The Tutoring Service creates a new instance of the Example Tracer in order to serve Fred's session and initializes it using information from the named Behavior Graph. It sends the initial problem values to the Flash movie so that the interface can display the initial problem state. After this, Fred can begin work on the problem.

As Fred performs problem-solving steps and asks for hints, each action is sent to his instance of the Example Tracer, which persists throughout the session. The Example Tracer evaluates student behavior by comparing it against the solution paths stored in the graph, as described above. The protocol between the student interface and tutor engine is described in [28]. Each student action and tutor response are logged to the *DataShop* whose rich logging format [44] permits extensive detail on the state of the tutor, including, for example, the multiple interpretations the tutor might have for a single student action. When the Example Tracer considers the problem finished, the Tutoring Service prepares and sends to the Flash movie a Problem Summary message with counts of hint use, correct and incorrect responses. The Flash move forwards the Problem Summary to the LMS, which stores the performance data for reporting and responds with information needed to run the next problem in the given problem set.

Fred's teacher Jane can use the *Mathtutor* report screens to track Fred's progress: these reports aggregate data provided in the Problem Summary messages just mentioned. In a coming version of the site, we plan to use skill information from these messages to choose problems for Fred adaptively, according to the "knowledge-tracing" algorithm in [19].

6.3 Architectural Choices

The remainder of this section motivates key choices underlying the *Mathtutor* architecture, which may be of general interest to the learning technologies community.

6.3.1 Choice #1: Running the Tutor Engine on the Server

As mentioned, a central architectural choice (inherited from CTAT) was to have a thin client, that is, to place the tutor engine on the server so that only minimal functionality runs on the client. Specifically, the tutor engine runs in a module called the *Tutoring Service,* a Java-language process on a Linux host. We see a number of pros of having the tutor engine run on the server, especially in an architecture such as CTAT's that aims to support not a single ITS, but many ITSs with a wide variety of tutor interfaces, implemented in any language, and can also be employed to provide tutoring within existing interfaces or simulators.

First, it is easier to combine the same tutor engine with different options for implementing student interfaces when the two communicate over sockets, unencumbered by difficulties associated with direct procedure calls between systems implemented in different programming languages. However, security restrictions in browsers typically permit socket connections only back to the Web application's host server, preventing tutor/interface interprocess communication locally on a client machine. Communication with a tutor engine on the server avoids this problem and still gains the ease of integration.

A second advantage of the thin client architecture is that it minimizes the requirements on the client machine and network so that the site is useful in schools and tutoring sites with older hardware. The client machines' processing and memory requirements are limited to those of the browser and the lightweight Flash Player. Download sizes are limited to the compiled Flash student interfaces, which tend to be small (10s or 100s of kilobytes). The remaining network usage consists of (frequent) short messages conveying student step descriptors and tutor evaluations; this usage pattern permits effective multiplexing of many concurrent sessions over low-bandwidth Internet connections, even if every student action in the tutor interface requires server access.

Third, placing the tutor engine on a server with full-time connectivity accommodates access by the tutor engine to a range of possible external resources (e.g., natural language processing, image analysis, computer algebra engines, etc.) that might not be suited to direct Web access from clients. This kind of interoperability is useful for possible future extensions of *Mathtutor* as well as for applications of the CTAT architecture outside of *Mathtutor.* External resources available via socket interfaces can be integrated by connections from the Tutoring Service, which is unrestricted by browser security regimes that inhibit client access to servers other than that which served the UI. The server-side tutor engine also facilitates access to external Java-based resources. They can be linked in as application libraries, without the need to download them to the client.

A key question is: Will a server-side tutor engine solution be able to serve potentially thousands of concurrent users without requiring large numbers of server machines? Our initial experience with approximately 60 simultaneous users in a school has been positive even with old server hardware (and we now have a brand-new server). But this matter remains to be seen: to get a measurement, we are developing load tests, which (more severe than real-world use) simulate many simultaneous student sessions with no pausing for think time between steps. As mentioned, our many-small-messages communication pattern is suited to low network bandwidth. Also, we continue to make the example-tracing algorithm more efficient. It already is considerably lighter weight than the more complex model tracer of CTAT.

6.3.2 Choice #2: Use of Flash for User Interfaces

CTAT has long supported both Flash and Java as implementation environments for tutor user interfaces. In the *Mathtutor* project, we opted to use Flash to implement the tutor interfaces for the *Mathtutor* site for a number of reasons. First, Flash interfaces are a convenient way of delivering complex tutor interfaces on the Web. The compact Flash Player is well supported across browsers and client machines. Second, Flash's ActionScript programming capabilities generally permit richer client functions than HTML- and Javascript-based interfaces, such as those found in other Web-based learning environments for mathematics, such as *Assistments* [46] and *ActiveMath* [32]. Third, Flash's frame and layer facilities enable easy creation of dynamic interfaces, described above. Finally, a version of Flash runs on handheld devices such as personal digital assistants (PDAs). Brown et al. [16] have implemented CTAT tutors on PDAs for the domain of college-level introductory business mathematics. Hence, it is conceivable that ultimately some of *Mathtutor's* problems can be run on the pocket-sized cell-enabled devices in growing use among middle- and high school students. However, there are some disadvantages to the use of Flash. The Flash integrated development environment (IDE) is not free. If we were to contemplate contributions from authors outside our lab, those authors would bear the cost of the tool. On the other hand, several Java IDEs (e.g., NetBeans, Eclipse) are available for free. Further, we are at the mercy of Adobe's development priorities; we have already discovered that backward compatibility of ActionScript versions appears not to be high priority for Adobe. On the whole, however, we feel that the advantages outweigh the disadvantages.

6.3.3 Choice #3: Use of Ruby on Rails to Implement Most of the Nontutoring Functionality

We opted to use Rails to implement functionality for student and class management, teacher reports, etc. Our choice followed the positive experience of the *Assistments* group with Rails. It offers several advantages. First, it affords fast prototyping and implementation for Web applications, with compact, easy-to-read code and lots of examples from the online development community. Second, integration with Java, which may be necessary for access to external resources, is straightforward. Third, with the Apache Web server and Rails' Mongrel server, it is easy to add multiple host machines for scalability [51].

6.3.4 Choice #4: Separation of Data for Normal Operation of the Site and for Research

The *Mathtutor* architecture strictly separates the data collected for research purposes from the data needed to support the normal operation of the site, mainly because the data collected for research purposes need to be kept in anonymous form (meaning that we remove all information from which the identity of students, teachers, and schools can be inferred), whereas the data needed for regular use should not be anonymous. For the purpose of research and development, detailed logs of step-by-step performance data are collected in the *DataShop*, where anonymization is automatic [26]. This facility was developed for the Pittsburgh Science of Learning Center (PSLC, http://www.learnlab.org/), a US National Science Foundation (NSF)-sponsored research center spanning Carnegie Mellon University and the University of Pittsburgh. The *DataShop* is a public available repository for many data sets collected by various kinds of educational technology, including many data sets from various tutors. It provides a suite of Web-based tools for analyzing these data and generates reports on error rates and learning curves, among other things. These reports differ from the teacher reports described above in that they are available in *DataShop* only, not on the *Mathtutor* site, and are geared more toward learning science researchers than teachers. These reports will be invaluable in the iterative redesign of the content on the site and also tell us how effective the tutors on the *Mathtutor* site are in helping students learn.

7 Comparison with Other Math Sites

There is a bewildering array of Web sites devoted to math instruction. Why then is there a need for *Mathtutor?* Many of the existing sites are of low quality, although the best offer useful services such as games to motivate students, expert help on problems submitted via the site, descriptions of concepts and procedures, examples, practice problems with answers, and online "interactive" textbooks. Some (but very few) existing Web sites provide guidance from software tutors. *Mathtutor* appears to be the only site with a (soon-to-be) comprehensive set of ITS for middle school mathematics, with step-by-step support on problems with multiple solution paths.

The *Assistments* Web site (http://www.assistment.org/) is the closest in concept and objective to *Mathtutor*. Like *Mathtutor,* the *Assistments* site provides online computer tutoring, complete with hints, immediate feedback, and dynamic scaffolding, and provides online reports for teachers [46]. Although the *Assistments* tutors derive from the same Cognitive Tutoring tradition as *Mathtutor*, there are some important distinctions. First, while *Assistments* is geared toward preparation for high-stakes standardized tests, *Mathtutor* will eventually support a comprehensive set of mathematics topics for the middle grades. Second, the underlying tutoring technology is different. *Assistments* supports simple tutors capable of recognizing only a single solution path for any given problem (Koedinger, personal communication), whereas example-tracing tutors support more sophisticated tutoring behaviors: they recognize multiple strategies within a problem, can deal with complex dependencies between problem steps, and can handle ambiguity about how any single student action should be interpreted. On the other hand, at the time of this writing, *Assistments* is a more mature site than *Mathtutor* and has demonstrated learning gains in empirical studies [46]. The large number of *Assistments* users is a strong indicator that there is a need for interactive Web-based computer tutoring in the area of K-12 mathematics.

Like *Mathtutor,* the *ActiveMath* mathematics Web site [32], [33] supports mathematics exercises with hints and feedback. However, *ActiveMath* aims to support a broader pedagogy than *Mathtutor*. It focuses on adaptive presentation of a range of instructional materials, and, more so than *Mathtutor,* it gives the student control over the instructional activities. *ActiveMath* is designed to be broadly applicable; so far, though, it has a limited curriculum of middle school math. Further, the math content on *ActiveMath,* although sophisticated in its use of AI technology, has not been empirically tested for learning benefits nearly to the extent that Cognitive Tutors, the basis of *Mathtutor*, have been.

Wayang Outpost [13] is an online system for high school mathematics aimed at high-stakes test preparation. It provides simple tutors embedded in a virtual adventure involving environmental science researchers stationed in Indonesia. These tutors provide multimedia hints. They do not, however, support complex, multistep problem solving—only multiple-choice solutions are supported. In *Animal Watch* [8], [18], mathematics problem solving is integrated with multimedia material about endangered species. *Animal Watch* targets middle school mathematics, although it does not appear to support a comprehensive curriculum. It provides limited support for step-by-step problem solving, with multimedia hints and interactive hints that illustrate strategies but do not show how to apply them to the problem at hand.

Among free sites for mathematics learning, the *Math Forum* Web site (http://mathforum.org) stands out due to the sheer scope and volume of materials and its many services. It provides a wide variety of resources, for a range of levels from basic math to advanced topics such as calculus. Within the "Dr. Math" section of the site, students can submit questions, which a group of volunteer teachers discuss and answer. The Dr. Math service, however, does not support interactive learning by doing; it merely turns a single problem instance into an example for a student to study. A "Math Tools" section contains many freely downloadable programs for students to use. While the number of programs is impressive (in the hundreds), most of the software appear to be lacking a strong pedagogical rationale, unlike the Cognitive Tutor approach used for the *Mathtutor* Web site.

There is also a multitude of commercial sites that provide services to math learners for a fee. As far as we have been able to ascertain, however, none of these sites provide access to ITS or provide step-by-step guidance with problem-solving exercises. Furthermore, a solid theoretical or empirical basis is typically lacking to back up the learning efficacy of the combination of services or materials that is offered on these sites. The textbook publisher *Prentice Hall* (http://www.phschool.com/math) presents "interactive online textbooks" for basic math, Prealgebra, Algebra 1, Algebra 2, and Geometry. These include audio clips that define terms and principles, worked examples for students to study, videos of math teachers explaining mathematical concepts and solving problems, and automatically graded multiple-choice questions. The materials, however, are not customizable and the site does not provide step-by-step guidance with problems. *Math.com* (http://www.math.com) sells materials and services for Prealgebra, Algebra, Geometry, Trigonometry, Statistics, and Calculus. The site contains mostly static practice and test materials that can be purchased through a yearly membership fee. *Math.com* also sells online tutoring provided by humans and delivered over the Internet for a hefty fee. *FirstInMath.com* (http://www.firstinmath.com) provides engaging math games, all using the same basic user interface, to motivate students to learn mathematics. It is based on the concept of a common fear—getting the wrong answer—by presenting problems in which the answer is given. However, *FirstInMath* does not provide step-by-step tutoring (only the final answer is checked), it is not customizable, and does not explicitly provide practice with multiple strategies. The *Compass-Learning Odyssey* Web site (http://www.childu.com) also strives to teach students by engaging them in mathematics game playing (as well as writing, reading, and social studies). It employs elaborate Flash-animated worked examples that include audio narration, followed by hands-on exercises. *CompassLearning* appears to emphasize rote mathematical learning of procedures.

There are many other Web sites, including *engrade* (http://engrade.com), which offers free Web-based tools for grading assignments, *AplusMath* (http://aplusmath.com), which provides many (untutored) exercises and games, and *SuperKids* (http://superkids.com), which automatically generates worksheets and answer sheets for teachers to use for tests or homework, but none of these sites is focused on computer-to-human tutoring, as is *Mathtutor*.

Finally, a recent review of mathematics e-Learning systems [58] cites over 30 systems, some of which provide ITS-like functionality, such as feedback and hints (e.g., *STACK* [53], *WALLIS* [30], and *ActiveMath,* discussed above). While this review provides much useful information, it focuses little attention on our greatest aim: How one can scale up a mathematics Web site for open-access use of intelligent tutors.

8 Discussion

The *Mathtutor* project brings together four strands of closely related ITS research as follows:

1. the Cognitive Tutor technology (e.g., [24]);
2. a set of Cognitive Tutor courses for middle school mathematics that was created in our lab;
3. a relatively new kind of ITS technology, *example-tracing tutors;* and
4. a set of authoring tools, CTAT, which makes it possible for computer-savvy nonprogrammers to create example-tracing tutors.

As we use CTAT's programming-by-demonstration approach to reimplement the existing Cognitive Tutors for middle school mathematics as example-tracing tutors, a key question addressed is whether programming by demonstration, by tech-savvy nonprogrammers is feasible for ITS development on a large scale. Put differently, can we have our cake and eat it too? Can we have simpler ITS authoring, more efficient tutor development, lower personnel cost, *as well as* large scale and maintainability? The viability of efficient approaches to ITS authoring is of broad interest to the ITS and learning technologies research communities, in light of the fact that ITS, while highly effective in improving student learning, have typically been difficult and time-consuming to build. Large-scale ITS projects such as *Mathtutor* are few and far between, so the *Mathtutor* project provides a unique opportunity for studying this issue. In this section, we discuss our initial experiences with respect to this scalability issue.

At this point in time, we are quite confident that the example-tracing technology will be able to support the reimplementation of the vast majority of the tutoring behaviors found within the existing middle school math tutors, supporting our claim that example-tracing tutors support many of the same tutoring behaviors that Cognitive Tutors provide. (As mentioned, our efforts in the early stages of the project are directed at reimplementing existing Cognitive Tutors as example-tracing tutors, in order to run them on the Web.) While example-tracing tutors are no doubt simpler than Cognitive Tutors, they are nonetheless able to support sophisticated tutoring behaviors. We, therefore, do not foresee that we will be watering down any of the tutors just to make them amenable to implementation as example-tracing tutors. We are also quite confident that computer-savvy nonprogrammers will be able to author tutors of sufficiently high quality. This confidence is based in part on our experience supporting CTAT users in earlier projects, reported in [1], in part on our experience in the *Mathtutor* project so far. We have been pleasantly surprised by how productive nonprogrammers have been with the CTAT tools. For instance, undergraduate students from CMU's School of Design have done most of the tutor development so far.

A key remaining question with respect to scalability is: Will tutors built through programming by demonstration be maintainable over an extended life cycle? It is tempting to think that because example-tracing tutors are (relatively) easy to create, they are also easy to maintain. But maintainability requires more than ease of creation/modification. A key requirement is that common structures (or code) are captured once and shared across contexts, rather than duplicated and copied for use in different contexts. In other words, a key question is whether a proliferation of behavior graphs that only differ in minor ways can be avoided within CTAT's programming-by-demonstration approach.

Fortunately, some of CTAT's advanced authoring features, in particular, formulas and Mass Production have the side effect that when used well, they enhance the maintainability of the resulting behavior graphs. First, the use of formulas tends to reduce the number of different paths in a behavior graph because the formula makes it possible to generalize across multiple solution paths, essentially collapsing different paths into one. No doubt, less elaborate graphs will be easier to maintain.

Second, the Mass Production facility makes it possible to author tutor behavior for many isomorphic or near-isomorphic tutor problems with a single template. It, therefore, helps dramatically in capturing, in a single place, behavior graph structure that is common across multiple tutor problems. Until recently, a limitation of the Mass Production facility was that the slightest variation in the problem space between two problems would preclude Mass Producing them using the same behavior graph template. We are, however, making progress in streamlining CTAT's Mass Production facility so that Mass Production can use the same behavior graph template for problems that are near isomorphs. We are taking advantage of a mechanism that was added for a different purpose, namely the ability to specify a lower and upper bound on the number of times a student should traverse a behavior graph link in order for the problem to be considered finished. By setting both limits to zero on a given link, this link is effectively removed from consideration by the example tracer. By including variables for the link traversal limits in a behavior graph template used for Mass Production, behavior graphs with and without the link (which, therefore, are near isomorphs) can be generated off of a single template, considerably enhancing the utility of Mass Production. Accommodating near isomorphs is important because in a realistic tutoring corpus such as the one for middle school math that we started with, we find many near-isomorphic problems. More generally, problem variability is desirable [42].

9 Conclusion

We have created an open-access Web site, called *Mathtutor,* where middle school students can learn mathematics with guidance from artificially intelligent software tutors.

Eventually, the site will provide a comprehensive set of tutors covering five key content strands for mathematics students in grades 6-8. The *Mathtutor* site is unique, even in the crowded landscape of Web sites for mathematics learning, because very few (if any) existing math sites support complex, multistrategy problem solving by means of ITS for a comprehensive range of middle school math topics. An important future goal of the project is to document the feasibility of the Web site in real educational settings. Although the Web site is designed to be useful in a large variety of contexts, our initial efforts have been to encourage adoption by teachers and schools with a focus on after-school tutoring programs. Detailed log data of student-tutor interactions on the site will help in evaluating whether *Mathtutor* helps students improve their mathematical competence.

The *Mathtutor* project is addressing an important open question in ITS and learning technologies research: Can ITS authoring be made simpler and more cost effective in a way that 1) supports sophisticated tutoring behaviors such as multiple solution strategies, dependencies among problem steps, and multiple interpretations of student behavior and 2) supports large-scale development, iterative refinement, and maintenance of ITS? We are employing a novel technology for ITS, called example-tracing tutors. These tutors maintain many of the key behaviors that make ITS effective, yet are based on a technology of programming by demonstration that makes them easier to build. Our experience in the *Mathtutor* project so far gives us confidence that a programming-by-demonstration approach is appropriate to support the range of tutoring behaviors needed for the middle school math tutors, and it will support iterative content development and refinement on a large scale. So far, nonprogrammers have been very productive using CTAT to develop example-tracing tutors. Further, we are confident that maintainability will not be an obstacle. CTAT's advanced authoring features, in particular, formulas and Mass Production, have the side effect that when used well, they enhance maintainability. A final issue regarding scale is whether the thin client approach taken in the *Mathtutor* architecture will support large numbers of simultaneous users. This issue is still open; we have laid out reasons that we think the thin client will work well.

The *Mathtutor* project will provide insight into whether open access to ITS for math learning, with an extremely low barrier for entry, is a useful way of enhancing math learning in a wide variety of settings. As evidence of the possibilities, sites such as *Assistments* [46] and *ActiveMath* [32], [33], which also have an intelligent tutoring math presence on the Web, have attracted many users. It is too early to tell whether *Mathtutor* will succeed equally admirably in this regard. We are hopeful, however, that *Mathtutor* will attract a large and diverse user population and contribute substantially to making ITS widespread.

Acknowledgments

This project is supported by the US Department of Education (IES) Award No. R305A080093.

References

[1] V. Aleven, B.M. McLaren, J. Sewall, and K.R. Koedinger, "Example-Tracing Tutors: A New Paradigm for Intelligent Tutoring Systems," *Int'l J. Artificial Intelligence and Education,* to appear.

[2] S.R. Alpert, M.K. Singley, and P.G. Fairweather, "Deploying Intelligent Tutors on the Web: An Architecture and an Example," *Int'l J. Artificial Intelligence in Education,* vol. 10, no. 2, pp. 183-197, 1999.

[3] J.R. Anderson, *Rules of the Mind.* Erlbaum, 1993.

[4] J.R. Anderson, A.T. Corbett, K.R. Koedinger, and R. Pelletier, "Cognitive Tutors: Lessons Learned," *J. Learning Sciences,* vol. 4, no. 2, pp. 167-207, 1995.

[5] J.R. Anderson and R. Pelletier, "A Development System for Model-Tracing Tutors," *Proc. Int'l Conf. Learning Sciences,* pp. 1-8, 1991.

[6] R.K. Atkinson, S.J. Derry, A. Renkl, and D. Wortham, "Learning from Examples: Instructional Principles from the Worked Examples Research," *Rev. of the Educational Research,* vol. 70, no. 2, pp. 181-214, 2000.

[7] R.K. Atkinson, A. Renkl, and M.M. Merrill, "Transitioning from Studying Examples to Solving Problems: Combining Fading with Prompting Fosters Learning," *J. Educational Psychology,* vol. 95, pp. 774-783, 2003.

[8] I. Arroyo, B.P. Woolf, and C.R. Beal, "Addressing Cognitive Differences and Gender during Problem Solving," *Technology, Instruction, Cognition, and Learning,* vol. 4, pp. 31-63, 2006.

[9] R.S. Baker, A.T. Corbett, and K.R. Koedinger, "Toward a Model of Learning Data Representations," *Proc. 23rd Ann. Conf. Cognitive Science Soc.,* pp. 45-50, 2001.

[10] R.S. Baker, A.T. Corbett, and K.R. Koedinger, "Learning to Distinguish between Representations of Data: A Cognitive Tutor That Uses Contrasting Cases," *Proc. Int'l Conf. Learning Sciences,* pp. 58-65, 2004.

[11] R.S.J.d. Baker, A.T. Corbett, and K.R. Koedinger, "The Difficulty Factors Approach to the Design of Lessons in Intelligent Tutor Curricula," *Int'l J. Artificial Intelligence in Education,* vol. 17, no. 4, pp. 341-369, 2007.

[12] R.S. Baker, A.T. Corbett, K.R. Koedinger, and M.P. Schneider, "A Formative Evaluation of a Tutor for Scatterplot Generation: Evidence on Difficulty Factors," *Proc. Conf. Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies (AI-ED '03),* U. Hoppe, F. Verdejo, and J. Kay, eds., pp. 107-114, 2003.

[13] C.R. Beal, R. Walles, I. Arroyo, and B.P. Woolf, "Online Tutoring for Math Achievement: A Controlled Evaluation," *J. Interactive Online Learning,* vol. 6, pp. 43-55, 2007.

[14] P. Black and D. William, "Inside the Black Box: Raising Standards through Classroom Assessment," *Phi Delta Kappan,* vol. 80, no. 2, pp. 139-148, http://www.pdkintl.org/kappan/kbla9810.htm, Oct. 1998.

[15] P. Black and D. William, "Assessment and Classroom Learning," *Assessment in Education,* vol. 5, no. 1, pp. 7-74, Mar. 1998.

[16] Q. Brown, F.J. Lee, D.D. Salvucci, and V. Aleven, "Interface Challenges for Mobile Tutoring Systems," *Proc. Ninth Int'l Conf. Intelligent Tutoring Systems (ITS '08),* B. Woolf, E. Aimeur, R. Nkambou, and S. Lajoie, eds., pp. 693-695, 2008.

[17] R. Catrambone, "The Subgoal Learning Model: Creating Better Examples so that Students Can Solve Novel Problems," *J. Experimental Psychology General,* vol. 127, pp. 355-376, 1998.

[18] P.R. Cohen, C.R. Beale, and N.M. Adams, "The Design, Deployment and Evaluation of the AnimalWatch Intelligent Tutoring System," *Proc. 18th European Conf. Artificial Intelligence (ECAI '08),* M. Ghallab, C.D. Spyropoulos, N. Fakotakis, and N. Avouris, eds., pp. 663-667, 2008.

[19] A.T. Corbett and J.R. Anderson, "Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge," *User Modeling and User-Adapted Interaction,* vol. 4, pp. 253-278, 1995.

[20] A. Corbett, A. Wagner, and J. Raspat, "The Impact of Analyzing Example Solutions on Problem Solving in a Pre-Algebra Tutor," *Artificial Intelligence in Education,* U. Hoppe, F. Verdejo and J. Kay, eds., *Proc. 11th Int'l Conf. Artificial Intelligence and Education (AIED '03),* pp. 133-140, 2003.

[21] N.T. Heffernan, T.E. Turner, A.L.N. Lourenco, M.A. Macasek, G. Nuzzo-Jones, and K.R. Koedinger, "The ASSISTment Builder: Towards an Analysis of Cost Effectiveness of ITS Creation," *Proc. Int'l Florida Artificial Intelligence Research Soc. Conf. (FLAIRS '06),* 2006.

[22] K.R. Koedinger and V. Aleven, "Exploring the Assistance Dilemma in Experiments with Cognitive Tutors," *Educational Psychology Rev.,* vol. 19, no. 3, pp. 239-264, 2007.

[23] K.R. Koedinger, V. Aleven, N. Heffernan, B.M. McLaren, and M. Hockenberry, "Opening the Door to Non-Programmers: Authoring Intelligent Tutor Behavior by Demonstration," *Proc. Seventh Int'l Conf. Intelligent Tutoring Systems (ITS '04),* J.C. Lester, R.M. Vicario, and F. Paraguaçu, eds., pp. 162-174, 2004.

[24] K.R. Koedinger, J.R. Anderson, W.H. Hadley, and M.A. Mark, "Intelligent Tutoring Goes to School in the Big City," *Int'l J. Artificial Intelligence in Education,* vol. 8, pp. 30-43, 1997.

[25] K.R. Koedinger and A.T. Corbett, "Cognitive Tutors: Technology Bringing Learning Science to the Classroom," *The Cambridge Handbook of the Learning Sciences,* K. Sawyer, ed., Cambridge Univ. Press, 2006.

[26] K. Koedinger, K. Cunningham, A. Skogsholm, and B. Leber, "An Open Repository and Analysis Tools for Fine-Grained, Longitudinal Learner Data," *Educational Data Mining 2008: Proc. First Int'l Conf. Educational Data Mining,* R.S.J.d., Baker, T. Barnes, and J.E. Beck, eds., pp. 157-166, 2008.

[27] K.R. Koedinger and A. Terao, "A Cognitive Task Analysis of Using Pictures to Support Pre-Algebraic Reasoning," *Proc. 24th Ann. Conf. Cognitive Science Soc.,* C.D. Schunn and W. Gray, eds., pp. 542-547, 2002.

[28] B. Leber et al., "CTAT's Tool-Tutor Application Interface," http://ctat.pact.cs.cmu.edu/index.php?id=tool-tutor, 2007.

[29] *Your Wish Is my Command: Programming by Example,* H. Lieberman ed. Morgan Kaufmann, 2001.

[30] M. Mavrikis and A. Maciocia, "WALLIS: A Web-Based ILE for Science and Engineering Students Studying Mathematics," *Proc. Workshop Advanced Technologies for Math. Education in 11th Int'l Conf. Artificial Intelligence in Education,* http://www.maths.ed.ac.uk/~wallis/, 2003.

[31] B.M. McLaren, S. Lim, and K.R. Koedinger, "When and How Often Should Worked Examples be Given to Students? New Results and a Summary of the Current State of Research," *Proc. 30th Ann. Conf. Cognitive Science Soc.,* B.C. Love, K. McRae, and V.M. Sloutsky, eds., pp. 2176-2181, 2008.

[32] E. Melis, E. Andrès, J. Büdenbender, A. Frischauf, G. Goguadze, P. Libbrecht, M. Pollet, and C. Ullrich, "ActiveMath: A Generic and Adaptive Web-Based Learning Environment," *Int'l J. Artificial Intelligence in Education,* vol. 12, pp. 385-407, 2001.

[33] E. Melis, G. Goguadze, M. Homik, P. Libbrecht, C. Ullrich, and S. Winterstein, "Semantic-Aware Components and Services of ActiveMath," *British J. Educational Technology,* vol. 37, no. 3, pp. 405-423, 2006.

[34] A. Mitrovic, P. Suraweera, B. Martin, K. Zakharov, N. Milik, and J. Holland, "Authoring Constraint-Based Tutors in ASPIRE," *Proc. Eighth Int'l Conf. Intelligent Tutoring Systems (ITS '06),* M. Ikeda, K.D. Ashley, and T.W. Chan, eds., pp. 41-50, 2006.

[35] A. Mitrovic, B. Martin, and P. Suraweera, "Intelligent Tutors for All: The Constraint-Based Approach," *IEEE Intelligent Systems,* vol. 22, no. 4, pp. 38-45, July/Aug. 2007.

[36] P. Morgan and S. Ritter, "An Experimental Study of the Effects of Cognitive Tutor Algebra I on Student Knowledge and Attitude," Carnegie Learning, Inc., http://www.carnegielearning.com, 2002.

[37] T. Murray, "An Overview of Intelligent Tutoring System Authoring Tools: Updated Analysis of the State of the Art," *Authoring Tools for Advanced Learning Environments,* T. Murray, S. Blessing, and S. Ainsworth, eds., chapter 17, pp. 491-544, Kluwer Academic Publishers, 2003.

[38] B.A. Myers, R.G. McDaniel, and D.S. Kosbie, "Marquise: Creating Complete User Interfaces by Demonstration," *Proc. Conf. Human Factors in Computing Systems (INTERCHI '93),* 1993.

[39] NCTM, *Curriculum Focal Points for Prekindergarten through Grade 8 Math.* Nat'l Council of Teachers of Math. 2008.

[40] NCTM, *Principles and Standards for School Math.* Nat'l Council of Teachers of Math., 2000.

[41] J. Ong and S. Noneman, "Intelligent Tutoring Systems for Procedural Task Training of Remote Payload Operations at NASA," *Proc. Industry/Interservice, Training, Simulation and Education Conf. (I/ITSEC '00),* 2000.

[42] F. Paas and J. Van Merriënboer, "Variability of Worked Examples and Transfer of Geometry Problem-Solving Skills: A Cognitive-Load Approach," *J. Educational Psychology,* vol. 86, pp. 122-133, 1994.

[43] J. Patvarczki, S.F. Almeida, J.E. Beck, and N.T. Heffernan, "Lessons Learned from Scaling Up a Web-Based Intelligent Tutoring System," *Proc. Ninth Int'l Conf. Intelligent Tutoring Systems (ITS '08),* B. Woolf, E. Aimeur, R. Nkambou, and S. Lajoie, eds., pp. 766-770, 2008.

[44] Pittsburgh Science of Learning Center, *Guide to the Tutor Message Format,* https://pslcdatashop.web.cmu.edu/dtd/guide/tutor_message_dtd_guide_v4.pdf, 2008.

[45] G.S. Plano, "The Effects of the Cognitive Tutor Algebra on Student Attitudes and Achievement in a Ninth Grade Algebra Course," PhD dissertation, Seton Hall Univ., unpublished.

[46] L. Razzaq, M. Feng, G. Nuzzo-Jones, N.T. Heffernan, K.R. Koedinger, B. Junker, S. Ritter, A. Knight, C. Aniszczyk, S. Choksey, T. Livak, E. Mercado, T.E. Turner, R. Upalekar, J.A. Walonoski, M.A. Macasek, and K.P. Rasmussen, "The Assistment Project: Blending Assessment and Assisting," *Proc. 12th Int'l Conf. Artificial Intelligence in Education,* C.K. Looi, G. McCalla, B. Bredeweg, and J. Breuker, eds., pp. 555-562, 2005.

[47] A. Renkl, "Worked-Out Examples: Instructional Explanations Support Learning by Self-Explanations," *Learning and Instruction,* vol. 12, no. 5, pp. 529-556, 2002.

[48] A. Renkl, R.K. Atkinson, and C.S. Große, "How Fading Worked Solution Steps Works—A Cognitive Load Perspective," *Instructional Science,* vol. 32, pp. 59-82, 2004.

[49] B. Rittle-Johnson and K.R. Koedinger, "Comparing Instructional Strategies for Integrating Conceptual and Procedural Knowledge," *Proc. Conf. Int'l Group for the Psychology of Math. Education, North Am. Chapter (PME-NA XXXIII),* 2002.

[50] B. Rittle-Johnson and K.R. Koedinger, "Using Cognitive Models to Guide Instructional Design: The Case of Fraction Division," *Proc. 23rd Ann. Conf. Cognitive Science Soc.,* pp. 857-862, 2001.

[51] S. Ruby, D. Thomas, and D.H. Hansson, *Agile Web Development with Rails,* third ed. The Pragmatic Programmers LLC, 2009.

[52] R. Salden, V. Aleven, A. Renkl, and R. Schwonke, "Worked Examples and Tutored Problem Solving: Redundant or Synergistic forms of Support?" *Proc. 30th Ann. Conf. Cognitive Science Soc.,* B.C. Love, K. McRae, and V.M. Sloutsky, eds., pp. 589-594, 2008.

[53] C. Sangwin and S. Hammond, "Enhancing Traditional Teaching through the STACK CAA System," *Proc. Fifth Workshop Joining Educational Math. (JEM),* http://www.stack.bham.ac.uk/, 2007.

[54] H. Sarkis, "Cognitive Tutor Algebra 1 Program Evaluation," Miami-Dade County Public Schools, The Reliability Group, Carnegie Learning, Inc., http://www.carnegielearning.com, 2004.

[55] R. Schwonke, A. Renkl, C. Krieg, J. Wittwer, V. Aleven, and R. Salden, "The Worked-Example Effect: Is It Just an Artefact of Lousy Control Conditions?" *Computers in Human Behavior,* to appear.

[56] J.G. Trafton and B.J. Reiser, "The Contributions of Studying Examples and Solving Problems to Skill Acquisition," *Proc. 15th Ann. Conf. Cognitive Science Soc.,* M. Polson, ed., pp. 1017-1022, 1993.

[57] K. VanLehn, "The Behavior of Tutoring Systems," *Int'l J. Artificial Intelligence in Education,* vol. 16, no. 3, pp. 227-265, 2006.

[58] WebALT "State of the Art in Mathematical E-Learning," EDC-22253-WEBALT, WebALT Consortium, WebALT Project Deliverable D1.1., http://webalt.math.helsinki.fi/content/e16/e301/e304/D1.1._State_of_the_Art_in_mathematical_e-learning_eng.pdf, Apr. 2005.

[59] G. Weber and P. Brusilovsky, "ELM-ART: An Adaptive Versatile System for Web-Based Instruction," *Int'l J. Artificial Intelligence in Education,* Special Issue on Adaptive and Intelligent Web-Based Educational Systems, vol. 12, no. 4, pp. 351-384, 2001.

[60] B.P. Woolf, *Building Intelligent Interactive Tutors: Student-Centered Strategies for Revolutionizing e-Learning.* Morgan Kaufmann, 2008.

[61] K. Yacef, "The Logic-ITA in the Classroom: A Medium Scale Experiment," *Int'l J. Artificial Intelligence in Education,* vol. 15, pp. 41-60, 2005.

Vincent Aleven is an assistant professor in the Human-Computer Interaction Institute at Carnegie Mellon University. He has 16 years of experience in research and development related to intelligent tutoring systems and authoring tools for tutoring systems. He is one of the original developers of the Cognitive Tutor Geometry curriculum. He has conducted research in real educational settings, in urban and suburban schools at the high school, middle school, and vocational school levels, as well as at the postsecondary level. He is a member of the executive committee of the Pittsburgh Science of Learning Center (http://www.learnlab.org).

Bruce M. McLaren has a split appointment as a senior systems scientist in the Human-Computer Interaction Institute at Carnegie Mellon University in Pittsburgh and as a senior researcher at the Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) in Saarbrücken, Germany. He has research interests in educational technology, collaborative learning, intelligent tutoring, and artificial intelligence. He has more than 50 publications in peer-reviewed journals, conferences, workshops, and symposiums, with most focused on educational technology and learning.

Jonathan Sewall is the project director in the Human-Computer Interaction Institute at Carnegie Mellon University. He has been the technical lead of the Cognitive Tutor Authoring Tools Project for the last four years. He has 25 years of experience in government, industry, and academia with software design and development. His past work experience includes expert systems, massively parallel systems, Web applications, databases, and networks.